DAVID R. EBERHART (S.B. #195474)
deberhart@omm.com
JAMES K. ROTHSTEIN (S.B. #267962)
jrothstein@omm.com
DANIEL H. LEIGH (S.B. #310673)
dleigh@omm.com
O'MELVENY & MYERS LLP
Two Embarcadero Center
28th Floor
San Francisco, California 94111-3823
Telephone: +1 415 984 8700
Facsimile: +1 415 984 8701

Attorneys for Plaintiffs
ELASTICSEARCH, INC. and
ELASTICSEARCH B.V.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

ELASTICSEARCH, INC., a Delaware corporation, and ELASTICSEARCH B.V., a Dutch corporation,

Plaintiffs,

v.

FLORAGUNN GmbH, a German corporation,

Defendant.

Case No.

COMPLAINT

1. COPYRIGHT INFRINGEMENT, 17 U.S.C. § 101 *ET SEQ*.
2. CONTRIBUTORY COPYRIGHT INFRINGEMENT

JURY TRIAL DEMAND

INTRODUCTION

1. Through its creation and distribution of its Search Guard software, floragunn GmbH ("floragunn") has knowingly and willfully infringed Elasticsearch, Inc. and elasticsearch B.V.'s (collectively "Elastic") copyrights in the source code for Elastic's Elasticsearch X-Pack and Kibana X-Pack software and their predecessors, Elasticsearch Shield and Kibana Shield. (Unless otherwise specified, Elastic refers to Shield and X-Pack collectively herein as "X-Pack.")

2. On September 4, 2019, Elastic brought an action against floragunn to remedy floragunn's infringement of certain Elastic copyrights in the source code for X-Pack. *See Elasticsearch, Inc. et al. v. floragunn GmbH*, Case No. 4:19-cv-05553-YGR (N.D. Cal.) ("*floragunn I*"). On November 26, 2019, Elastic filed a First Amended Complaint ("FAC") in the *floragunn I* lawsuit, alleging additional instances of copyright infringement and identifying additional Elastic X-Pack copyrights infringed by floragunn.

3. In the course of subsequent discovery and investigation, Elastic has identified yet more instances of infringement by floragunn and additional Elastic X-Pack copyrights that floragunn has infringed. Elastic has now registered each of those additional copyrights with the United States Copyright Office.

4. Elastic files this new lawsuit in light of recent Northern District of California decisions interpreting 17 U.S.C. § 411(a). *See* Order re: Joint Motion for Clarification, ECF No. 59, *UAB "Planner 5D" v. Facebook, Inc.*, No. 19-cv-03132-WHO (N.D. Cal. March 5, 2020); *Izmo, Inc. v. Roadster, Inc.*, No. 18-cv-06092-NC, 2019 WL 2359228 (N.D. Cal. June 4, 2019). Elastic will seek relation of this case to *floragunn I* pursuant to Civil Local Rule 3-12 and consolidation with *floragunn I* for all purposes.

PARTIES

5. Plaintiff Elasticsearch, Inc. is incorporated in Delaware; it has its principal place of business in Mountain View, California. Plaintiff elasticsearch B.V. is incorporated in the Netherlands.

6. Defendant floragunn is a German company with a principal place of business in Berlin, Germany.

JURISDICTION AND VENUE

7. Elastic's claims for copyright infringement arise under the Copyright Act of 1976, 17 U.S.C. § 101 *et seq*.

8. This Court has original subject matter jurisdiction of this action under 28 U.S.C. §§ 1331 and 1338.

9. This Court has specific personal jurisdiction over floragunn because, among other reasons, floragunn has extensively offered and distributed its infringing product containing Elastic's copyrighted material to companies in California and purposefully committed within California the acts from which Elastic's claims arise. Additionally, to the extent floragunn has committed the illegal acts described herein outside of California, it did so knowing and intending that such acts would cause injury to Elastic at its principal place of business within California.

10. Venue is proper in the Northern District of California under 28 U.S.C. § 1391(b)(2) and 1391(c)(3) because a substantial part of the events or omissions giving rise to the claims alleged in this complaint occurred in this judicial district.

INTRADISTRICT ASSIGNMENT

11. Because this action arises from Elastic's assertion of its intellectual property rights, Northern District of California Civil Local Rule 3-2(c) excludes this action from the division-specific venue rule and subjects this action to assignment on a district-wide basis.

THE ELASTIC STACK AND X-PACK SOFTWARE

12. Elastic produces a core suite of search and analytics products known as the Elastic Stack (formerly known as ELK Stack). The Elastic Stack consists of Elasticsearch, Logstash, Kibana, and Beats. Elasticsearch is a search and analytics engine. Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch. Kibana lets users visualize data with charts and graphs in Elasticsearch. Beats is a family of "data shipper" software that collects and centralizes data that feeds into the other products in Elastic Stack.

13. X-Pack is a set of add-on features to Elastic's core Elastic Stack suite of products. X-Pack includes security, altering, monitoring, reporting, and other add-ons to Elasticsearch,

Kibana, and other products in the Elastic Stack. The predecessor to much of X-Pack was known as Shield. (Unless otherwise specified, Elastic refers to Shield and X-Pack collectively herein as "X-Pack.")

14. Elastic has a longstanding commitment to opening the source code underlying many of its products in order to facilitate building a community that helps improve and advance Elastic's products to produce the best software possible. When Elastic releases the source code for its software, it does so under clearly delineated conditions.

15. In late April 2018, Elastic opened the source code for version 6.2.x of X-Pack. Elastic made the code available on Elastic's public GitHub code repository for users to inspect, contribute, create issues, and open pull requests, all pursuant to the "Elastic License." Elastic has released the source code for subsequent versions of X-Pack on GitHub, also under the "Elastic License."

16. The Elastic License did not grant to floragunn or any other party the right to create copies or prepare derivative works for use in any production capacity. And to the extent floragunn acquired any rights pursuant to the Elastic License, those rights terminated immediately and automatically by virtue of floragunn's actions as described herein. Nor did any license applicable to earlier versions of X-Pack and/or Shield provide floragunn the right to create copies or prepare derivative works for use in any production capacity.

17. Elastic is informed and believes, and, on that basis, alleges that floragunn accessed the Elastic code described herein either through decompilation of Elastic binaries, reviewing source-available Elastic repositories, and/or review of otherwise publicly-available Elastic code.

FLORAGUNN'S INFRINGEMENT OF ELASTIC'S COPYRIGHTS IN X-PACK

18. floragunn markets and distributes Search Guard, a plug-in for Elasticsearch that offers features similar to the security features that Elastic offers in X-Pack. floragunn makes certain source code for Search Guard available for review and inspection on its GitLab repositories under several different license agreements.

19. Search Guard is available as a "Community Edition" for free for certain uses, but floragunn charges customers for Enterprise and Compliance editions of Search Guard. floragunn

prohibits users from, among other things, taking features from the Enterprise or Compliance editions of Search Guard into production without purchasing a license. In fact, floragunn explicitly warned its users that doing so "is illegal" and "can lead to serious legal consequences, which can bring more harm and costs to a company . . . ."

FLORAGUNN'S INFRINGEMENT OF ELASTIC'S COPYRIGHTS IN X-PACK

20. After initiating the *floragunn I* lawsuit, Elastic identified further instances of infringement by floragunn. Infringement by floragunn is evident in at least the following code from a February 13, 2016 commit to the Search Guard PrivilegesEvaluator.java file:

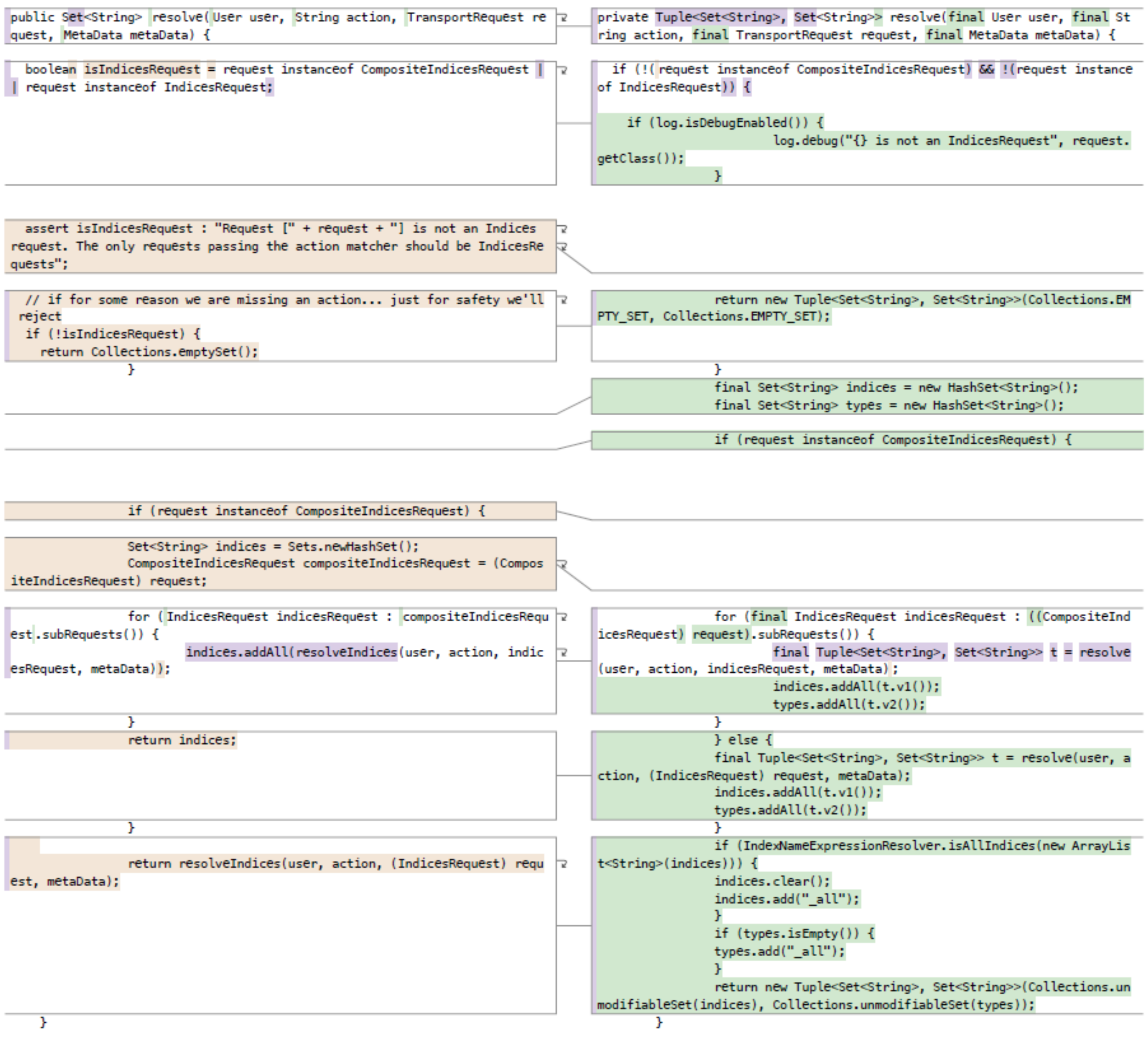
```
private Tuple<Set<String>, Set<String>> resolve(final User user, final String action, final
TransportRequest request,
                final MetaData metaData) {

        if (!(request instanceof CompositeIndicesRequest) && !(request instanceof
IndicesRequest)) {

                if (log.isDebugEnabled()) {
                log.debug("{} is not an IndicesRequest", request.getClass());
                }

                return new Tuple<Set<String>, Set<String>>(Collections.EMPTY_SET,
Collections.EMPTY_SET);
        }
        final Set<String> indices = new HashSet<String>();
        final Set<String> types = new HashSet<String>();
        if (request instanceof CompositeIndicesRequest) {
                for (final IndicesRequest indicesRequest : ((CompositeIndicesRequest)
request).subRequests()) {
                final Tuple<Set<String>, Set<String>> t = resolve(user, action,
indicesRequest, metaData);
                indices.addAll(t.v1());
                types.addAll(t.v2());
                }
        } else {
                final Tuple<Set<String>, Set<String>> t = resolve(user, action,
(IndicesRequest) request, metaData);
                indices.addAll(t.v1());
                types.addAll(t.v2());
        }
        if (IndexNameExpressionResolver.isAllIndices(new ArrayList<String>(indices)))
{
                indices.clear();
                indices.add("_all");
        }
        if (types.isEmpty()) {
                types.add("_all");
        }
        return new Tuple<Set<String>,
Set<String>>(Collections.unmodifiableSet(indices), Collections.unmodifiableSet(types));
        }
```

21. The code in Paragraph 20 is copied from and/or is a derivative work of at least the following Elastic code included in the binary of Elasticsearch Shield in the file DefaultIndicesResolver.java, original to Elasticsearch Shield version 1.0.0 Beta1 and released by Elastic on November 3, 2014:

```
    public Set<String> resolve(User user, String action, TransportRequest request,
MetaData metaData) {

        boolean isIndicesRequest = request instanceof CompositeIndicesRequest || request
instanceof IndicesRequest;
        assert isIndicesRequest : "Request [" + request + "] is not an Indices request. The
only requests passing the action matcher should be IndicesRequests";

        // if for some reason we are missing an action... just for safety we'll reject
        if (!isIndicesRequest) {
                return Collections.emptySet();
        }

        if (request instanceof CompositeIndicesRequest) {
                Set<String> indices = Sets.newHashSet();
                CompositeIndicesRequest compositeIndicesRequest =
(CompositeIndicesRequest) request;
                for (IndicesRequest indicesRequest :
compositeIndicesRequest.subRequests()) {
                indices.addAll(resolveIndices(user, action, indicesRequest, metaData));
                }
                return indices;
        }

        return resolveIndices(user, action, (IndicesRequest) request, metaData);
    }
```

22. Elastic registered Elasticsearch Shield version 1.0.0 Beta1 with the United States Copyright Office on April 30, 2020.

23. As shown below, ignoring non-substantive differences in the code and adjusting white space to illustrate similarities, it is clear that at least the floragunn code in Paragraph 20 (on the right) is copied from and/or is a derivative work of at least the Elastic code in Paragraph 21 (on the left):



```
public Set<String> resolve(User user, String action, TransportRequest re
quest, MetaData metaData) {
  boolean isIndicesRequest = request instanceof CompositeIndicesRequest |
| request instanceof IndicesRequest;

  assert isIndicesRequest : "Request [" + request + "] is not an Indices
request. The only requests passing the action matcher should be IndicesRe
quests";

  // if for some reason we are missing an action... just for safety we'll
 reject
  if (!isIndicesRequest) {
    return Collections.emptySet();
                }

                if (request instanceof CompositeIndicesRequest) {

                Set<String> indices = Sets.newHashSet();
                CompositeIndicesRequest compositeIndicesRequest = (Compos
iteIndicesRequest) request;

                for (IndicesRequest indicesRequest : compositeIndicesRequ
est.subRequests()) {
                        indices.addAll(resolveIndices(user, action, indic
esRequest, metaData));

                }
                return indices;

                }

                return resolveIndices(user, action, (IndicesRequest) requ
est, metaData);
    }
```

```
private Tuple<Set<String>, Set<String>> resolve(final User user, final St
ring action, final TransportRequest request, final MetaData metaData) {
  if (!(request instanceof CompositeIndicesRequest) && !(request instance
of IndicesRequest)) {

    if (log.isDebugEnabled()) {
                        log.debug("{} is not an IndicesRequest", request.
getClass());
                }

                return new Tuple<Set<String>, Set<String>>(Collections.EM
PTY_SET, Collections.EMPTY_SET);

                }
                final Set<String> indices = new HashSet<String>();
                final Set<String> types = new HashSet<String>();

                if (request instanceof CompositeIndicesRequest) {

                for (final IndicesRequest indicesRequest : ((CompositeInd
icesRequest) request).subRequests()) {
                        final Tuple<Set<String>, Set<String>> t = resolve
(user, action, indicesRequest, metaData);
                        indices.addAll(t.v1());
                        types.addAll(t.v2());
                }
                } else {
                final Tuple<Set<String>, Set<String>> t = resolve(user, a
ction, (IndicesRequest) request, metaData);
                indices.addAll(t.v1());
                types.addAll(t.v2());
                }
                if (IndexNameExpressionResolver.isAllIndices(new ArrayLis
t<String>(indices))) {
                indices.clear();
                indices.add("_all");
                }
                if (types.isEmpty()) {
                types.add("_all");
                }
                return new Tuple<Set<String>, Set<String>>(Collections.un
modifiableSet(indices), Collections.unmodifiableSet(types));
        }
```

24. Infringement by floragunn is also evident in an April 5, 2017 commit to the Search Guard Kibana plugin login_controller.js file. That file contains the following code:

```
const {query, hash} = parse($window.location.href, true);
let nextUrl;
if (query.next) {
    nextUrl = query.next + (hash || '')
} else {
    nextUrl = "/";
}
```

25. That floragunn code is copied from and/or is a derivative work of the following Elastic code included in parse_next.js and original to Kibana Shield version 2.3.2 released by Elastic on April 26, 2016, reproduced here:

```
const {query, hash} = parse(location.href, true);
if (query.next) return query.next + (hash || '');
return '/';
```

26. Elastic registered Kibana Shield version 2.3.2 with the United States Copyright Office on November 6, 2019.

27. Infringement by floragunn is also evident in an August 6, 2017 commit to the Search Guard Kibana plugin index.html and filterbar.html files. Those files contain the following code:

filterbar.html Lines 2–32	`<div class="kuiToolBar">`
	`<div class="kuiToolBarSearch">`
	`<div class="kuiToolBarSearchBox">`
	`<div class="kuiToolBarSearchBox__icon kuiIcon fa-search"></div>`
	`<input class="kuiToolBarSearchBox__input ng-pristine ng-untouched ng-valid" type="text" placeholder="Search..." ng-model="query">`
	`</div>`
	`</div>`
	
	`<div class="kuiToolBarSection">`
	`<a ng-click="new()" class="kuiButton kuiButton--primary kuiButton--iconText">`
	`<span class="kuiButton__icon kuiIcon fa-plus"></span>`
	`Add`
	`</a>`
	
	`<a ng-href="#/" class="kuiButton kuiButton--basic kuiButton--iconText">`
	`<span class="kuiButton__icon kuiIcon fa-chevron-left"></span>`
	`Back`
	`</a>`
	
	`</div>`
	
	`<div class="kuiToolBarSection">`
	`<!-- We need an empty section for the buttons to be positioned consistently. -->`
	`</div>`
	`</div>`
	`<!-- NoResults -->`
	`<div class="kuiPanel kuiPanel--centered ng-hide" ng-show="!(resourcenames | filter:query).length">`
	`<div class="kuiNoItems">`

	No `<span ng-show="query" class="ng-hide">matching</span>` entries found. `</div>` `</div>`
index.html Line 13	`<table class="kuiTable" ng-show="(resourcenames | filter:query).length">`
index.html Line 34	`<tr ng-repeat="username in resourcenames | filter:query" data-test-subj="userRow" class="kuiTableRow">`
index.html Lines 44–48	`<td class="kuiTableRowCell" data-test-subj="userRowRoles">` `<div class="kuiTableRowCell__liner">` `<div ng-repeat="role in resources[username].roles">{{role}}<br/></div>` `</div>` `</td>`

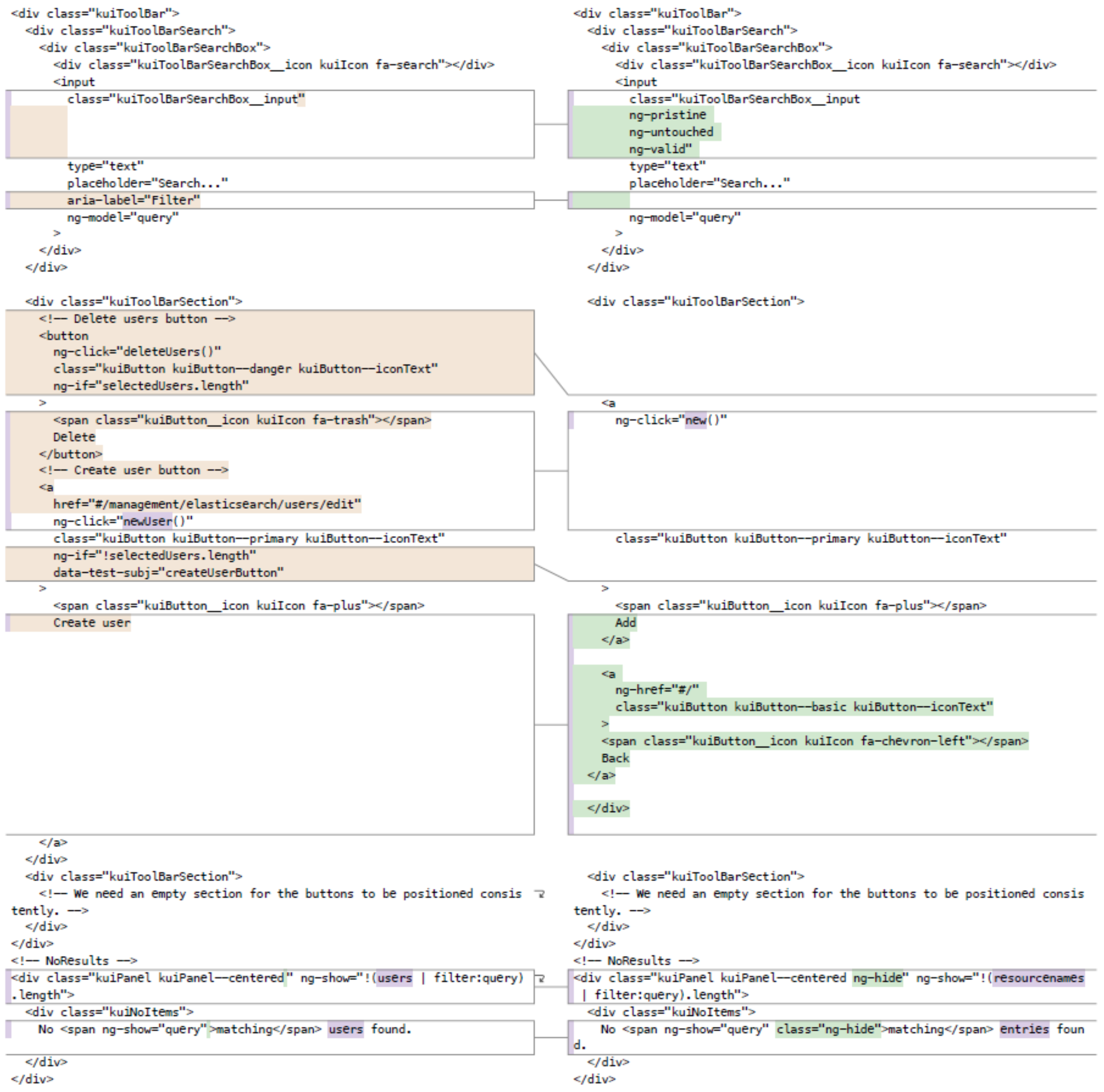
28. That floragunn code is copied from and/or is a derivative work of Elastic code in the users.html file original to Kibana X-Pack versions 5.0.0 and 5.2.0 and released by Elastic on or before January 31, 2017, reproduced here:

Lines 22–70	`<div class="kuiToolBar">` `<div class="kuiToolBarSearch">` `<div class="kuiToolBarSearchBox">` `<div class="kuiToolBarSearchBox__icon kuiIcon fa-search"></div>` `<input` `class="kuiToolBarSearchBox__input"` `type="text"` `placeholder="Search..."` `aria-label="Filter"` `ng-model="query"` `>` `</div>` `</div>` `<div class="kuiToolBarSection">` `<!-- Delete users button -->` `<button` `ng-click="deleteUsers()"` `class="kuiButton kuiButton--danger kuiButton--iconText"` `ng-if="selectedUsers.length"` `>` `<span class="kuiButton__icon kuiIcon fa-trash"></span>` `Delete` `</button>` `<!-- Create user button -->` `<a` `href="#/management/elasticsearch/users/edit"` `ng-click="newUser()"` `class="kuiButton kuiButton--primary kuiButton--iconText"` `ng-if="!selectedUsers.length"` `data-test-subj="createUserButton"`

	> <span class="kuiButton__icon kuiIcon fa-plus"></span> Create user </a> </div> <div class="kuiToolBarSection"> <!-- We need an empty section for the buttons to be positioned consistently. --> </div> </div> <!-- NoResults --> <div class="kuiPanel kuiPanel--centered" ng-show="!(users | filter:query).length"> <div class="kuiNoItems"> No <span ng-show="query">matching</span> users found. </div> </div>
Line 73	<table class="kuiTable" ng-show="(users | filter:query).length">
Lines 135–39	<tr ng-repeat="user in users | orderBy:'username' | filter:query | orderBy:sort.orderBy:sort.reverse" data-test-subj="userRow" class="kuiTableRow" >
Lines 176–82	<td class="kuiTableRowCell" data-test-subj="userRowRoles"> <div class="kuiTableRowCell__liner"> <span ng-repeat="role in user.roles"> <a class="kuiLink" ng-href="#/management/elasticsearch/roles/edit/{{role}}">{{ role }}</a><span ng-if="!$last">,</span> </span> </div> </td>

29. As shown below, ignoring non-substantive differences in the code and adjusting white space to illustrate similarities, it is clear that at least the floragunn code in Paragraph 27 is copied from and/or is a derivative work of at least the Elastic code in Paragraph 28. This comparison includes additional indicia of copying in the following non-functional comment: “<!-- We need an empty section for the buttons to be positioned consistently. -->”

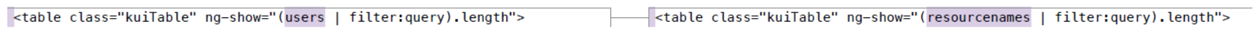
(a) filterbar.html lines 2–32 (on the right) and users.html lines 22–70 (on the left):



```
<div class="kuiToolBar">
  <div class="kuiToolBarSearch">
    <div class="kuiToolBarSearchBox">
      <div class="kuiToolBarSearchBox__icon kuiIcon fa-search"></div>
      <input
        class="kuiToolBarSearchBox__input"
        type="text"
        placeholder="Search..."
        aria-label="Filter"
        ng-model="query"
      >
    </div>
  </div>

  <div class="kuiToolBarSection">
    <!-- Delete users button -->
    <button
      ng-click="deleteUsers()"
      class="kuiButton kuiButton--danger kuiButton--iconText"
      ng-if="selectedUsers.length"
    >
      <span class="kuiButton__icon kuiIcon fa-trash"></span>
      Delete
    </button>
    <!-- Create user button -->
    <a
      href="#/management/elasticsearch/users/edit"
      ng-click="newUser()"
      class="kuiButton kuiButton--primary kuiButton--iconText"
      ng-if="!selectedUsers.length"
      data-test-subj="createUserButton"
    >
      <span class="kuiButton__icon kuiIcon fa-plus"></span>
      Create user
    </a>
  </div>
  <div class="kuiToolBarSection">
    <!-- We need an empty section for the buttons to be positioned consis
tently. -->
  </div>
</div>
<!-- NoResults -->
<div class="kuiPanel kuiPanel--centered" ng-show="!(users | filter:query)
.length">
  <div class="kuiNoItems">
    No <span ng-show="query">matching</span> users found.
  </div>
</div>
```

```
<div class="kuiToolBar">
  <div class="kuiToolBarSearch">
    <div class="kuiToolBarSearchBox">
      <div class="kuiToolBarSearchBox__icon kuiIcon fa-search"></div>
      <input
        class="kuiToolBarSearchBox__input
        ng-pristine
        ng-untouched
        ng-valid"
        type="text"
        placeholder="Search..."
        ng-model="query"
      >
    </div>
  </div>

  <div class="kuiToolBarSection">
    <a
      ng-click="new()"
      class="kuiButton kuiButton--primary kuiButton--iconText"
    >
      <span class="kuiButton__icon kuiIcon fa-plus"></span>
      Add
    </a>

    <a
      ng-href="#/"
      class="kuiButton kuiButton--basic kuiButton--iconText"
    >
    <span class="kuiButton__icon kuiIcon fa-chevron-left"></span>
    Back
  </a>

  </div>
  <div class="kuiToolBarSection">
    <!-- We need an empty section for the buttons to be positioned consis
tently. -->
  </div>
</div>
<!-- NoResults -->
<div class="kuiPanel kuiPanel--centered ng-hide" ng-show="!(resourcenames
 | filter:query).length">
  <div class="kuiNoItems">
    No <span ng-show="query" class="ng-hide">matching</span> entries foun
d.
  </div>
</div>
```

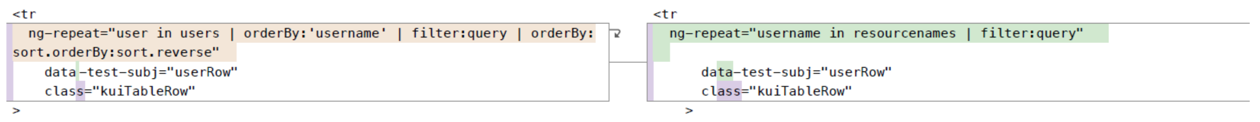
(b) index.html line 13 (on the right) and users.html line 73 (on the left):



```
<table class="kuiTable" ng-show="(users | filter:query).length">
```

```
<table class="kuiTable" ng-show="(resourcenames | filter:query).length">
```

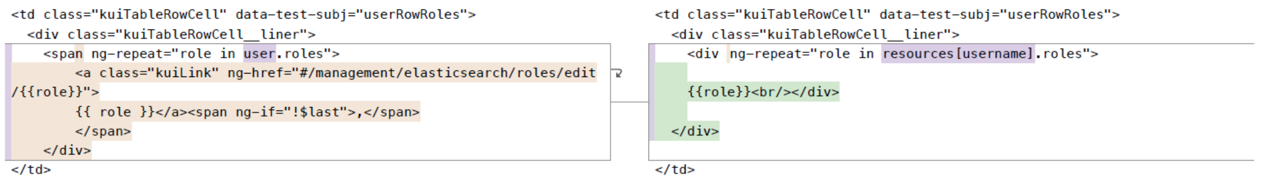
(c) index.html Line 34 (on the right) and users.html Lines 135–139 (on the left):



```
<tr
  ng-repeat="user in users | orderBy:'username' | filter:query | orderBy:
sort.orderBy:sort.reverse"
    data-test-subj="userRow"
    class="kuiTableRow"
>
```

```
<tr
  ng-repeat="username in resourcenames | filter:query"

    data-test-subj="userRow"
    class="kuiTableRow"
>
```

(d) index.html Lines 44–48 (on the right) and users.html Lines 176–182 (on the left):



```
<td class="kuiTableRowCell" data-test-subj="userRowRoles">
  <div class="kuiTableRowCell__liner">
    <span ng-repeat="role in user.roles">
        <a class="kuiLink" ng-href="#/management/elasticsearch/roles/edit
/{{role}}">
        {{ role }}</a><span ng-if="!$last">,</span>
        </span>
    </div>
</td>
```

```
<td class="kuiTableRowCell" data-test-subj="userRowRoles">
  <div class="kuiTableRowCell__liner">
    <div ng-repeat="role in resources[username].roles">

    {{role}}<br/></div>

  </div>
</td>
```

30. Elastic registered Kibana X-Pack versions 5.0.0 and 5.2.0 with the United States Copyright Office on April 30, 2020 and September 19, 2019, respectively.

31. Further newly identified infringement by floragunn is evident in a September 11, 2017 commit to the Search Guard PrivilegesEvaluator.java file, including at least:

```
if(action.equals("indices:data/write/bulk[s]")) {
```

and the following commented-out—that is, non-functional—code:

```
if(request instanceof BulkRequest) {

        for(DocWriteRequest<?> ar: ((BulkRequest) request).requests()) {
        //require also op type permissions
        switch(ar.opType()) {
        case CREATE: additionalPermissionsRequired.add(IndexAction.NAME);break;
        case INDEX: additionalPermissionsRequired.add(IndexAction.NAME);break;
        case DELETE: additionalPermissionsRequired.add(DeleteAction.NAME);break;
        case UPDATE: additionalPermissionsRequired.add(UpdateAction.NAME);break;
        }

        }
  }
```

32. A subsequent September 26, 2017 commit to the Search Guard PrivilegesEvaluator.java file contains a further example of floragunn's infringement. In that commit, floragunn retained the infringing commented-out code quoted in Paragraph 31, but removed the "/*" and "*/" symbols that had commented out the infringing code.

33. Then, floragunn again infringed Elastic's copyrights in an October 1, 2017 commit to the Search Guard PrivilegesEvaluator.java file. That commit added the following code:

```
switch (bir.request().opType()) {
case CREATE:
additionalPermissionsRequired.add(IndexAction.NAME);
break;
```

```
case INDEX:
additionalPermissionsRequired.add(IndexAction.NAME);
break;
case DELETE:
additionalPermissionsRequired.add(DeleteAction.NAME);
break;
case UPDATE:
additionalPermissionsRequired.add(UpdateAction.NAME);
break;
```

The same floragunn commit and file also commented out the following code:

```
/*if(request instanceof BulkRequest) {

for(DocWriteRequest<?> ar: ((BulkRequest) request).requests()) {
//require also op type permissions
switch(ar.opType()) {
case CREATE: additionalPermissionsRequired.add(IndexAction.NAME);break;
case INDEX: additionalPermissionsRequired.add(IndexAction.NAME);break;
case DELETE: additionalPermissionsRequired.add(DeleteAction.NAME);break;
case UPDATE: additionalPermissionsRequired.add(UpdateAction.NAME);break;
}

}
}*/
```

34. An October 10 commit to the same file demonstrates further infringement by floragunn. That commit removed the commented-out infringing code in Paragraph 33 but retained other infringing code. Additionally, floragunn's October 10 commit contains code that is a derivative work of Elastic's copyrighted source code.

35. The floragunn commits referenced in Paragraphs 31–34 contain code that is copied from and/or is a derivative work of at least the following Elastic code included in X-Pack in the file AuthorizationService.java, original to Elasticsearch X-Pack 5.6.0 and released by Elastic on September 11, 2017:

```
if (action.equals(TransportShardBulkAction.ACTION_NAME)) {
```

And:

```
final DocWriteRequest docWriteRequest = item.request();
switch (docWriteRequest.opType()) {
case INDEX:
case CREATE:
return IndexAction.NAME;
case UPDATE:
return UpdateAction.NAME;
case DELETE:
```

```
return DeleteAction.NAME;
}
```

36. Elastic had not publicly released the source code for X-Pack alleged in Paragraph 35 at the time of floragunn's copying and/or creation of derivative works from that code. Elastic is informed and believes and, on that basis, alleges that floragunn decompiled Elastic's binaries or otherwise gained access to Elastic's source code to create the copies and/or derivative works referenced in Paragraphs 31–34. Further, Elastic is informed and believes and, on that basis, alleges that the decompilation process employed by floragunn caused the code originally written by Elastic as:

```
if (action.equals(TransportShardBulkAction.ACTION_NAME)) {
```

to be rendered as:

```
if(action.equals("indices:data/write/bulk[s]")) {
```

37. Elastic is informed and believes and, on that basis, alleges that the creation and inclusion of the decompiled code referenced in Paragraphs 31 and 36 is both an act of infringement and evidence of infringing acts by floragunn.

38. Elastic registered Elasticsearch X-Pack 5.6.0 with the United States Copyright Office on October 13, 2020.

39. After initiating the *floragunn I* lawsuit, Elastic identified further infringement by floragunn in a June 7, 2018 commit to Search Guard in the file DlsFlsValveImpl.java. Infringement by floragunn is evident in at least the following code from the Search Guard DlsFlsValveImpl.java file:

```
if(request instanceof BulkShardRequest) {
        for(BulkItemRequest inner:((BulkShardRequest) request).items()) {
                if(inner.request() instanceof UpdateRequest) {
                listener.onFailure(new ElasticsearchSecurityException("Update is not
supported when FLS or DLS is activated"));
                return false;
                }
        }
}
```

40. The floragunn code in Paragraph 39 is copied from and/or is a derivative work of at least the following Elastic code included in the file BulkShardRequestInterceptor.java, original to Elasticsearch X-Pack version 5.1.1 and released on December 8, 2016:

```
        for (BulkItemRequest bulkItemRequest : request.items()) {
            IndicesAccessControl.IndexAccessControl indexAccessControl =
indicesAccessControl.getIndexPermissions(bulkItemRequest.index());
            if (indexAccessControl != null) {
                boolean fls =
indexAccessControl.getFieldPermissions().hasFieldLevelSecurity();
                boolean dls = indexAccessControl.getQueries() != null;
                if (fls || dls) {
                    if (bulkItemRequest.request() instanceof UpdateRequest) {
                        throw new ElasticsearchSecurityException("Can't execute a bulk request
with update requests embedded if " +
                                "field or document level security is enabled",
RestStatus.BAD_REQUEST);
                    }
                }
            }
            logger.trace("intercepted bulk request for index [{}] without any update requests,
continuing execution",
                    bulkItemRequest.index());
        }
    }

    @Override
    public boolean supports(TransportRequest request) {
        return request instanceof BulkShardRequest;
    }
}
```

41. Elastic registered Elasticsearch X-Pack version 5.1.1 with the United States Copyright Office on October 13, 2020.

42. Further infringement by floragunn is evident in at least the following commented-out code from the June 7, 2018 commit to the Search Guard DlsFlsValveImpl.java file:

```
aliasRequest.getAliasActions().stream().filter(a->a.actionType() ==
Type.ADD).forEach(a->{
```

And:

```
listener.onFailure(new ElasticsearchSecurityException("Managing aliases is not supported
when FLS or DLS is activated"));
```

43. The floragunn code in Paragraph 42 is copied from and/or is a derivative work of at least the following Elastic code included the file IndicesAliasesRequestInterceptor.java, released December 19, 2017 and original to Elasticsearch X-Pack version 6.1.1:

```
for (IndicesAliasesRequest.AliasActions aliasAction : request.getAliasActions()) {
if (aliasAction.actionType() == IndicesAliasesRequest.AliasActions.Type.ADD) {
```

And:

```
throw new ElasticsearchSecurityException("Alias requests are not allowed for users who
have " +
"field or document level security enabled on one of the indices",
RestStatus.BAD_REQUEST);
```

44. Elastic registered Elasticsearch X-Pack version 6.1.1 with the United States Copyright Office on October 13, 2020.

45. Further infringement by floragunn is evident in a June 4, 2019 commit to the Search Guard searchguard_saved_objects_client.js file. That file contains the following code:

```
const ACTIONS = {
  CREATE: 'create',
  BULK_CREATE: 'bulk_create',
  FIND: 'find',
  GET: 'get',
  BULK_GET: 'bulk_get', // @todo Why the snake case here? What do our permissions
look like
  UPDATE: 'update',
  'DELETE': 'delete',
};
export default class SearchguardSavedObjectsClient {
  constructor(request, searchguardBackend, requestRepository, savedObjects) {
      this.request = request;
      this.searchguardBackend = searchguardBackend;
      this.callWithRequestRepository = requestRepository;
      this.errors = savedObjects.SavedObjectsClient.errors;
      this.savedObjectTypes = savedObjects.types;
  }
  async create(type, attributes = {}, options = {}) {
      return await this.authorize(
        type,
        ACTIONS.CREATE,
        repository => repository.create(type, attributes, options)
      );
  }
  async bulkCreate(objects, options = {}) {
      let types = [];
      objects.forEach((object) => {
        if (types.indexOf(object.type) === -1) {
          types.push(object.type);
        }
      });

      return await this.authorize(
        types,
        ACTIONS.BULK_CREATE,
        repository => repository.bulkCreate(objects, options)
      );
    }
    async find(options = {}) {
      return await this.authorize(
        options.type,
        ACTIONS.FIND,
        repository => repository.find(options)
      );
    }
    async get(type, id) {
      return await this.authorize(
        type,
        ACTIONS.GET,
        repository => repository.get(type, id)
      );
    }
    async bulkGet(objects = []) {
      let types = [];
      objects.forEach((object) => {
        if (types.indexOf(object.type) === -1) {
          types.push(object.type);
        }
      });
      return await this.authorize(
        types,
        ACTIONS.BULK_GET,
        repository => repository.bulkGet(objects)
      );
    }
    async update(type, id, attributes, options = {}) {
      return await this.authorize(
        type,
        ACTIONS.UPDATE,
        repository => repository.update(type, id, attributes, options)
      )
    }
    async delete(type, id) {
      return await this.authorize(
        type,
        ACTIONS['DELETE'],
        repository => repository.delete(type, id)
      );
    }
    buildPermissionName(action, type) {
      return `kibana:saved_objects/${type}/${action}`;
    }
```

```
  getPermissionInfo(type, action) {
    const types = Array.isArray(type) ? type : [type];
    let permissionToType = {};
    const permissionNames = types.map((type) => {
      const permissionName = this.buildPermissionName(action, type);
      permissionToType[permissionName] = type;
      return permissionName;;
    });

    return {
      types,
      permissionNames,
      permissionToType,
      permissionsParameter: permissionNames.join(',')
    };
  }
  async authorize(type, action, clientCallback) {
    const permissionsInfo = this.getPermissionInfo(type, action);
    const permissionsResult = await
this.checkPermissions(permissionsInfo.permissionsParameter,
permissionsInfo.permissionToType);

    if (permissionsResult.hasAllPermissions !== true) {
      const errorMessage = `Unauthorized: ${action} for ${permissionsInfo.types.join(',
')}, missing permissions: ${permissionsResult.missingPermissions.join(', ')}`;
      // @todo Error handling in the frontend
      throw this.errors.decorateForbiddenError(new Error(errorMessage));
    } else {
      return await clientCallback(this.callWithRequestRepository);
    }
  }
  async checkPermissions(permissionsParameter, permissionToType) {
    try {
      const backendResult = await
this.searchguardBackend.hasPermissions(this.request.headers, permissionsParameter);

      let checkResult = {
        hasAllPermissions: false,
        missingPermissions: [],
        missingTypes: [],
        allowedTypes: [],
      };

      // Go through the response for each permission
      for (let permission in backendResult.permissions) {
        let permissionType = permissionToType[permission];
        if (backendResult.permissions[permission] !== true) {
          checkResult.missingPermissions.push(permission);
          checkResult.missingTypes.push(permissionType);
        } else if (backendResult.permissions[permission] === true) {
          checkResult.allowedTypes.push(permissionType)
        }
      }

      if (checkResult.missingPermissions.length) {
```

```
                checkResult.hasAllPermissions = false;
                return checkResult;
              } else {
                checkResult.hasAllPermissions = true;
                return checkResult;
              }

          } catch (error) {
              throw this.errors.decorateGeneralError(error, 'Could not check for application
permissions');
          }
        }
      }
```

46. As further alleged below, at least substantial portions of that floragunn code is copied from and/or is a derivative work of following Elastic code released in a July 24, 2018 commit to secure_saved_objects_client.js and original to Kibana X-Pack 6.4.0, reproduced here:

```
export class SecureSavedObjectsClient {
  constructor(options) {
    const {
      errors,
      internalRepository,
      callWithRequestRepository,
      checkPrivileges,
      auditLogger,
      savedObjectTypes,
      actions,
    } = options;
    this.errors = errors;
    this._internalRepository = internalRepository;
    this._callWithRequestRepository = callWithRequestRepository;
    this._checkPrivileges = checkPrivileges;
    this._auditLogger = auditLogger;
    this._savedObjectTypes = savedObjectTypes;
    this._actions = actions;
  }
  async create(type, attributes = {}, options = {}) {
    return await this._execute(
      type,
      'create',
      { type, attributes, options },
      repository => repository.create(type, attributes, options),
    );
  }
  async bulkCreate(objects, options = {}) {
    const types = uniq(objects.map(o => o.type));
    return await this._execute(
      types,
      'bulk_create',
      { objects, options },
      repository => repository.bulkCreate(objects, options),
    );
  }
  async delete(type, id) {
    return await this._execute(
      type,
      'delete',
      { type, id },
      repository => repository.delete(type, id),
    );
  }
  async find(options = {}) {
    if (options.type) {
      return await this._findWithTypes(options);
    }

    return await this._findAcrossAllTypes(options);
  }
  async bulkGet(objects = []) {
    const types = uniq(objects.map(o => o.type));
    return await this._execute(
      types,
      'bulk_get',
      { objects },
      repository => repository.bulkGet(objects)
    );
  }
  async get(type, id) {
    return await this._execute(
      type,
      'get',
      { type, id },
      repository => repository.get(type, id)
    );
  }
  async update(type, id, attributes, options = {}) {
    return await this._execute(
      type,
      'update',
      { type, id, attributes, options },
      repository => repository.update(type, id, attributes, options)
    );
  }
  async _checkSavedObjectPrivileges(actions) {
    try {
      return await this._checkPrivileges(actions);
    } catch(error) {
      const { reason } = get(error, 'body.error', {});
      throw this.errors.decorateGeneralError(error, reason);
    }
  }
  async _execute(typeOrTypes, action, args, fn) {
    const types = Array.isArray(typeOrTypes) ? typeOrTypes : [typeOrTypes];
    const actions = types.map(type => this._actions.getSavedObjectAction(type, action));
    const { result, username, missing } = await this._checkSavedObjectPrivileges(actions);
    switch (result) {
        case CHECK_PRIVILEGES_RESULT.AUTHORIZED:
          this._auditLogger.savedObjectsAuthorizationSuccess(username, action, types, args);
          return await fn(this._internalRepository);
        case CHECK_PRIVILEGES_RESULT.LEGACY:
          return await fn(this._callWithRequestRepository);
        case CHECK_PRIVILEGES_RESULT.UNAUTHORIZED:
          this._auditLogger.savedObjectsAuthorizationFailure(username, action, types,
missing, args);
          const msg = `Unable to ${action} ${[...types].sort().join(',')}, missing
${[...missing].sort().join(',')}`;
          throw this.errors.decorateForbiddenError(new Error(msg));
        default:
          throw new Error('Unexpected result from hasPrivileges');
      }
    }
  async _findAcrossAllTypes(options) {
    const action = 'find';
    // we have to filter for only their authorized types
    const types = this._savedObjectTypes;
    const typesToPrivilegesMap = new Map(types.map(type => [type,
this._actions.getSavedObjectAction(type, action)]));
    const { result, username, missing } = await
this._checkSavedObjectPrivileges(Array.from(typesToPrivilegesMap.values()));
    if (result === CHECK_PRIVILEGES_RESULT.LEGACY) {
      return await this._callWithRequestRepository.find(options);
    }
    const authorizedTypes = Array.from(typesToPrivilegesMap.entries())
      .filter(([ , privilege]) => !missing.includes(privilege))
      .map(([type]) => type);
    if (authorizedTypes.length === 0) {
      this._auditLogger.savedObjectsAuthorizationFailure(
        username,
        action,
        types,
        missing,
        { options }
      );
      throw this.errors.decorateForbiddenError(new Error(`Not authorized to find
saved_object`));
    }
    this._auditLogger.savedObjectsAuthorizationSuccess(username, action,
authorizedTypes, { options });
    return await this._internalRepository.find({
      ...options,
      type: authorizedTypes
    });
  }
  async _findWithTypes(options) {
    return await this._execute(
      options.type,
      'find',
      { options },
      repository => repository.find(options)
    );
  }
```

}

47. As shown below, adjusting the order of the code in Paragraphs 45 and 46 to illustrate similarities, it is clear that at least the floragunn code shown below (on the right) is copied from and/or is a derivative work of at least the Elastic code shown below (on the left):

export class SecureSavedObjectsClient {
 	
constructor(options) {
const {
errors,
internalRepository,
callWithRequestRepository,
checkPrivileges,
auditLogger,
savedObjectTypes,
actions,
} = options;
	export default class SearchguardSavedObjectsClient {
	constructor(request, searchguardBackend, requestRepository, savedObjects) {
	this.request = request;
	this.searchguardBackend = searchguardBackend;
	this.callWithRequestRepository = requestRepository;
	this.errors = savedObjects.SavedObjectsClient.errors;
	this.savedObjectTypes = savedObjects.types;
	}

async create(type, attributes = {}, options = {}) {
return await this._execute(
type,
'create',
{ type, attributes, options },
repository => repository.create(type, attributes, options),
);
}
	async create(type, attributes = {}, options = {}) {
	return await this.authorize(
	type,
	ACTIONS.CREATE,
	
	repository => repository.create(type, attributes, options)
	);
	}

async bulkCreate(objects, options = {}) {
const types = uniq(objects.map(o => o.type));
 	
return await this._execute(
types,
	async bulkCreate(objects, options = {}) {
	let types = [];
	objects.forEach((object) => {
	if (types.indexOf(object.type) === -1) {
	types.push(object.type);
	}
	});
	
	return await this.authorize(
	types,

`'bulk_create',`
`{ objects, options },`
`repository =>`
`repository.bulkCreate(objects,`
`options),`
`);`
`}`
	`ACTIONS.BULK_CREATE,`
	
	`repository =>`
	`repository.bulkCreate(objects, options)`
	`);`
	`}`

`async delete(type, id) {`
`return await this._execute(`
`type,`
`'delete',`
`{ type, id },`
`repository =>`
`repository.delete(type, id),`
`);`
`}`
	`async delete(type, id) {`
	`return await this.authorize(`
	`type,`
	`ACTIONS['DELETE'],`
	
	`repository =>`
	`repository.delete(type, id)`
	`);`
	`}`

`async find(options = {}) {`
`if (options.type) {`
`return await`
`this._findWithTypes(options);`
`}`
 	
`return await`
`this._findAcrossAllTypes(options);`
`}`
	`async find(options = {}) {`
	
	`return await this.authorize(`
	`options.type,`
	`ACTIONS.FIND,`
	`repository =>`
	`repository.find(options)`
	`);`
	`}`

`async bulkGet(objects = []) {`
`const types = uniq(objects.map(o =>`
`o.type));`
 	
`return await this._execute(`
`types,`
`'bulk_get',`
`{ objects },`
`repository =>`
`repository.bulkGet(objects)`
`);`
`}`
	`async bulkGet(objects = []) {`
	`let types = [];`
	`objects.forEach((object) => {`
	`if (types.indexOf(object.type)`
	`=== -1) {`
	`types.push(object.type);`
	`}`
	`});`
	`return await this.authorize(`
	`types,`
	`ACTIONS.BULK_GET,`
	
	`repository =>`
	`repository.bulkGet(objects)`
	`);`

	`}`
`async get(type, id) { return await this._execute( type, 'get', { type, id }, repository => repository.get(type, id) ); }`	`async get(type, id) { return await this.authorize( type, ACTIONS.GET, repository => repository.get(type, id) ); }`
`async update(type, id, attributes, options = {}) { return await this._execute( type, 'update', { type, id, attributes, options }, repository => repository.update(type, id, attributes, options) ); }`	`async update(type, id, attributes, options = {}) { return await this.authorize( type, ACTIONS.UPDATE, repository => repository.update(type, id, attributes, options) ) }`

48. Elastic registered Kibana X-Pack version 6.4.0 with the United States Copyright Office on September 21, 2019.

49. Elastic also previously alleged multiple additional instances of infringement of its copyrights in X-Pack and the Kibana X-Pack plugin in its original complaint and FAC in *floragunn I*.

50. As alleged in *floragunn I*, infringement by floragunn is also evident in a February 13, 2016 commit to the file SearchGuardFilter.java. That commit and file contains commented out code in at least lines 48–52 that is copied from or is, at least, a derivative work of Elastic code in the file Privilege.java that is original to Elasticsearch Shield versions 1.0.0 Beta1, 1.1.1, and 2.0.0-beta1 and was released in or before 2015. Elastic registered Elasticsearch Shield version

1.0.0 Beta1 with the United States Copyright Office on April 30, 2020 and registered versions 1.1.1 and 2.0.0-beta1 on September 10, 2019.

51. As alleged in *floragunn I*, infringement by floragunn is evident in a May 30, 2016 commit to the SearchGuardSSLNettyHttpServerTransport.java file in at least lines 67–88. That code is copied from and/or is a derivative work of Elastic code included in the file ShieldNettyHttpServerTransport.java that was released on June 24, 2015 and is original to Elasticsearch Shield version 1.3.0. Elastic registered Elasticsearch Shield version 1.3.0 with the United States Copyright Office on September 10, 2019.

52. As alleged in *floragunn I*, infringement by floragunn is evident in a June 6, 2016 commit to the Search Guard file FlsFilterLeafReader.java in the search-guard-module-dlsfls repository in at least lines 165–177. That code is copied from and/or is a derivative work of Elastic code in the file FieldSubsetReader.java that was released by Elastic on September 17, 2015 and is original to Elasticsearch Shield version 2.0.0-beta2. Elastic registered Elasticsearch Shield version 2.0.0-beta2 with the United States Copyright Office on September 11, 2019.

53. As alleged in *floragunn I*, infringement by floragunn is evident in a March 31, 2018 commit to the Search Guard get_next_url.js file in least lines 21–42. That code is copied from and/or is a derivative work of code that Elastic included in a bug fix to the Kibana X-Pack plugin in parse_next.js file and that was released on or before April 20, 2017 and is original to Kibana X-Pack version 5.3.1 and Kibana Shield version 2.3.2. Elastic registered Kibana X-Pack version 5.3.1 and Kibana Shield version 2.3.2 with the United States Copyright Office on September 19, 2019 and November 6, 2019, respectively.

54. As alleged in *floragunn I*, infringement by floragunn is evident in a June 7, 2018 commit to the file DlsFlsFilterLeafReader.java, where floragunn copied the implementations of at least two methods, getLiveDocs and numDocs, from X-Pack. floragunn's implementation of getLiveDocs and numDocs in the June 7, 2018 commit to DlsFlsFilterLeafReader.java, found at least in lines 403–473 of that file, is copied from and/or is a derivative work of Elastic's implementation of getLiveDocs and numDocs in the file DocumentSubsetReader.java that Elastic made source-available for the first time in late April 2018 as part of Elasticsearch X-Pack version

6.2.x. This infringed Elastic code is original to Elasticsearch Shield version 2.0.0 RC1, which was released on October 7, 2015 and which Elastic registered with the United States Copyright Office on April 30, 2020.

55. As alleged in *floragunn I*, infringement by floragunn is evident in an October 28, 2018 commit to the Search Guard get_next_url.js file in at least lines 22–44. That code is copied from and/or is a derivative work of code that Elastic included in a bug fix to the Kibana X-Pack plugin in parse_next.js and that was released on or before January 30, 2018 and is original to Kibana X-Pack versions 5.6.7 and 5.3.1 and Kibana Shield version 2.3.2. Elastic registered Kibana X-Pack versions 5.6.7 and 5.3.1 and Kibana Shield version 2.3.2 with the United States Copyright Office on September 21, 2019, September 19, 2019, and November 6, 2019, respectively.

56. As alleged in *floragunn I*, infringement by floragunn is evident in an August 30, 2019 commit to the Search Guard file call_with_request_factory.js in at least lines 1–13. That code is copied from and/or is a derivative work of Elastic code that occurs multiple places within the Kibana X-Pack plugin, including in a February 28, 2019 commit to call_with_request_factory.js, and that is original to Kibana X-Pack versions 5.4.0 and 7.2.0. Elastic registered Kibana X-Pack versions 5.4.0 and 7.2.0 with the United States Copyright Office on November 5, 2019.

57. As alleged in *floragunn I*, infringement by floragunn is evident in an August 30, 2019 commit to the Search Guard fetch_all_from_scroll.js file in at least lines 1–19. That code is copied from and/or is a derivative work of Elastic code included in fetch_all_from_scroll.js that was released on May 4, 2017 and is original to Kibana X-Pack version 5.4.0. Elastic registered Kibana X-Pack version 5.4.0 with the United States Copyright Office on November 5, 2019.

<u>**FLORAGUNN INDUCES THIRD PARTIES TO INFRINGE ELASTIC'S COPYRIGHTS**</u>

58. floragunn's marketing and distribution of infringing Search Guard software causes third party Search Guard customers and users to incorporate code that infringes Elastic's copyrights in X-Pack and the Kibana X-Pack plugin. Those third parties therefore necessarily reproduce and use Elastic's proprietary X-Pack and/or Kibana X-Pack plugin code when they

incorporate Search Guard into their adoptions of Elasticsearch, thereby infringing Elastic's copyrights.

59. Additional third parties have incorporated floragunn's infringing code into products and services they offer publicly. Elastic has investigated to identify third parties who have incorporated floragunn's infringing code into their products and services.

60. Infringing third party products and services include at least Amazon.com, Inc.'s and Amazon Web Services Inc.'s Open Distro for Elasticsearch ("Open Distro") and Amazon Elasticsearch Service ("AESS"), which both contain and/or contained infringing code that originated with floragunn. Open Distro contains and/or contained infringing code that arises from floragunn's infringement of Elastic's copyrights in X-Pack and the Kibana X-Pack plugin. AESS contains or contained infringing code that arises from floragunn's infringement of Elastic's copyrights in X-Pack. Rackspace US, Inc.'s ObjectRocket for Elasticsearch contains or contained infringing code that arises from floragunn's infringement of Elastic's copyrights in X-Pack and the Kibana X-Pack plugin. And IBM Corporation's IBM Cloud Databases for Elasticsearch contains or contained infringing code that arises from floragunn's infringement of Elastic's copyrights in X-Pack.

<u>FIRST CAUSE OF ACTION</u>

Copyright Infringement

(17 U.S.C. § 101 *et seq.*)

61. Elastic incorporates by reference each of the allegations in the preceding paragraphs of this Complaint as if fully set forth here.

62. As alleged above, on August 14, 2019, Elastic registered with the United States Copyright Office versions 1.0.0 and 2.0.0 of Elasticsearch Shield and versions 5.0.0, 6.0.0, 6.2.0, 6.2.x, and 6.3.0 of X-Pack under Registration Numbers TX 8-762-996, TX 8-762-994, TX 8-762-975, TX 8-762-985, TX 8-762-987, TX 8-762-988, and TX 8-762-991, respectively. On September 10, 2019, Elastic registered versions 1.1.1, 1.3.0, 2.0.0-beta1, and, on September 11, 2019, 2.0.0-beta2 of Elasticsearch Shield under Registration Numbers TX 8-773-254, TX 8-773-258, TX 8-773-261, and TX 8-773-263, respectively. Elastic additionally registered: version 2.3.2

of the Kibana Shield plugin on November 6, 2019 under Registration Number TX 8-796-945; versions 5.2.0 and 5.3.1 of the Kibana X-Pack plugin on September 19, 2019 under Registration Numbers TX 8-777-406 and TX 8-777-412, respectively; and versions 5.6.7 and 6.4.0 of the Kibana X-Pack plugin on September 21, 2019 under Registration Numbers TX 8-778-023 and TX 8-778-024, respectively; version 5.4.0 of the Kibana X-Pack plugin on November 5, 2019 under Registration Number TX 8-796-010; and version 7.2.0 of the Kibana X-Pack plugin under Registration Number TX 8-796-013 on November 5, 2019. Elastic also registered, on April 30, 2020, Elasticsearch Shield versions 1.0.0 Beta1 and 2.0.0 RC1 and Kibana X-Pack version 5.0.0 under Registration Numbers TXu 2-191-552, TX 8-865-693, and TX 8-865-685, respectively.[1] On October 13, 2020, Elastic further registered version 2.0.1 of Elasticsearch Shield and versions 5.1.1, 5.6.0, and 6.1.1 of Elasticsearch X-Pack under Registration Numbers TX 8-902-217, TX 8-902-221, TX 8-902-228, and TX 8-902-227, respectively. Copies of those Certificates of Registration are attached as Exhibits A through Y to this Complaint.[2]

63. These works contain copyrightable subject matter for which copyright protection exists under the Copyright Act, 17 U.S.C. § 101, *et seq*. elasticsearch B.V. is the exclusive owner of all rights in these copyrighted works. Elasticsearch, Inc. holds the exclusive license from elasticsearch B.V. to enforce the copyright in and distribute copies of these works in, among other territories, the United States.

64. Through the actions described herein and in the *floragunn I* complaint and FAC, floragunn has infringed and will continue to infringe Elastic's copyrights in the X-Pack and Kibana X-Pack plugin code by, at least, reproducing (including through creation of intermediate

[1] Elasticsearch Shield version 1.0.0 Beta1 was registered as unpublished due to its release as a limited beta version.

[2] Exhibits W–Y are unofficial certificate previews for Elasticsearch Shield versions 1.0.0 Beta1 and 2.0.0 RC1 and Kibana X-Pack version 5.0.0. Due to the COVID-19 pandemic, the United States Copyright Office temporarily ceased creating and mailing official Certificates of Registration, and Elastic has not received official Certificates of Registration for these works. However, the United States Copyright Office has approved registration of these works, and their registration status may be verified through the United States Copyright Office's website, https://www.copyright.gov/.

copies), preparing derivative works from (including through decompilation), and distributing copies of those copyrighted works. No license permitted floragunn's infringing activities.

65. floragunn's infringing conduct alleged herein was and continues to be willful and with full knowledge of Elastic's rights in the copyrighted works, and that conduct has enabled floragunn to profit illegally from infringement.

66. Elastic is entitled to an injunction restraining floragunn, its officers, agents, employees, assigns, and all persons acting in concert with them from engaging in further infringement of Elastic's copyrights.

67. Elastic is entitled to recover from floragunn the damages it has sustained and will sustain as a result of floragunn's wrongful acts as alleged herein. Elastic is further entitled to recover from floragunn the gains, profits, and advantages it has obtained as a result of floragunn's wrongful acts. The full extent of Elastic's damages and the gains, profits, and advantages floragunn has obtained by reason of its aforesaid acts of copyright infringement cannot be determined at this time, but will be proven at trial. Further, Elastic is entitled to recover costs and reasonable attorneys' fees from floragunn as a result of the wrongful acts alleged herein.

<u>SECOND CAUSE OF ACTION</u>

Contributory Copyright Infringement

68. Elastic incorporates by reference each of the allegations in the preceding paragraphs of this Complaint as if fully set forth here.

69. floragunn's distribution of infringing Search Guard software induces, causes, encourages, and materially contributes to Search Guard users and third parties that incorporate Search Guard code into their products and services infringing Elastic's copyrights in the X-Pack and/or Kibana X-Pack plugin code by engaging in unauthorized reproduction and distribution of works containing Elastic's copyrighted material.

70. Elastic is informed and believes, and, on that basis, alleges that floragunn derived substantial financial benefit from Search Guard users' and third parties' infringement of Elastic's copyrights in X-Pack and/or the Kibana X-Pack plugin.

71. floragunn's marketing, commercial distribution of, licensing of, and profit from infringing Search Guard software shows that it knowingly, intentionally, willfully, and purposefully induced, caused, encouraged, and materially contributed to, and continues to knowingly, intentionally, willfully, and purposefully induce, cause, encourage, and materially contributes to, Search Guard users' and third parties' infringement of Elastic's copyrights in X-Pack and/or the Kibana X-Pack plugin.

72. floragunn has the ability to prevent Search Guard users and third parties from infringing Elastic's copyrights in the X-Pack and Kibana X-Pack plugin code by omitting the infringing code from its Search Guard software product. However, floragunn has not prevented Search Guard users and third parties from infringing Elastic's copyrights in the X-Pack and Kibana X-Pack plugin code.

73. floragunn, through its knowing and intentional inducement, causation, encouragement, and material contribution to the infringement of Elastic's copyrights in the X-Pack and Kibana X-Pack plugin code by Search Guard users and third parties, is committing and/or is contributorily and vicariously liable for the acts of infringement by Search Guard users and third parties. Each act of infringement that floragunn knowingly and intentionally induced, caused, encouraged, and materially contributed to is a separate and distinct act of infringement.

74. Elastic is entitled to an injunction restraining floragunn, its officers, agents, employees, assigns, and all persons acting in concert with them from actions inducing, causing, encouraging, or materially contributing to Search Guard users' and third parties' infringement of Elastic's copyrights.

75. Elastic is entitled to recover from floragunn the damages it has sustained and will sustain as a result of floragunn's acts inducing, causing, encouraging, or materially contributing to Search Guard users' and third parties' infringement of Elastic's copyrights. Elastic is further entitled to recover from floragunn the gains, profits, and advantages it has obtained as a result of its acts inducing, causing, encouraging, or materially contributing to Search Guard users' and third parties' infringement of Elastic's copyrights. The full extent of Elastic's damages and the gains, profits, and advantages floragunn has obtained by reason of its aforesaid acts of copyright

infringement by Search Guard users and third parties cannot be determined at this time but will be proven at trial. Further, Elastic is entitled to recover costs and reasonable attorneys' fees from floragunn as a result of the acts inducing, causing, encouraging, or materially contributing to Search Guard users' and third parties' infringement of Elastic's copyrights alleged herein.

PRAYER FOR RELIEF

Elastic prays for judgment as follows:

1. For permanent injunctive relief, including an order restraining and enjoining floragunn and third parties using Search Guard and Search Guard code from further infringement of Elastic's copyrights, specifically:

   a. that floragunn and third parties using Search Guard products and code, as well as any successor entities, directors and officers, agents, servants, employees, assigns, and all other persons acting in active concert or privity or in participation with them, and each of them, be enjoined from continuing to market, offer, sell, dispose of, license, lease, transfer, display, advertise, reproduce, develop or manufacture infringing Search Guard software and any works derived or copied from infringing Search Guard software, or to participate or assist in any such activity;

   b. that floragunn and third parties using Search Guard products and code, as well as any successor entities, directors and officers, agents, servants, employees, assigns, and all other persons acting in active concert or privity or in participation with them, be enjoined from directly or indirectly infringing Elastic's copyrights in X-Pack and the Kibana X-Pack plugin;

   c. that floragunn and third parties using Search Guard products and code, as well as any successor entities, directors and officers, agents, servants, employees, assigns, and all other persons acting in active concert or privity or in participation with them, be enjoined to return to Elastic any originals, copies, facsimiles, or duplicates of Search Guard, any works derived or copied from

Search Guard in their possession, custody, or control that are shown to infringe any Elastic copyright;

d. that floragunn and third parties using Search Guard products and code be enjoined to deliver upon oath, to be impounded during the pendency of this action, and for destruction pursuant to judgment herein, all originals, copies, facsimiles, or duplicates of Search Guard, any works derived or copied from Search Guard in their possession, custody, or control that are shown to infringe any Elastic copyright;

2. For compensatory damages against floragunn in an amount to be determined at trial;

3. For floragunn's profits obtained as a result of its infringing conduct, including but not limited to all profits from sales and other exploitation of Elastic's copyrighted material and any products, works, or other materials that include, copy, are derived from, or otherwise embody the copyrighted material, or in the Court's discretion, such amount as the Court finds to be just and proper;

4. For attorneys' fees and costs of suit incurred herein;

5. For interest, including pre-judgment and post-judgment interest, on the forgoing sums; and

6. For any other relief that the Court deems appropriate.

JURY DEMAND

Elastic demands a jury trial for all issues so triable.

Dated: October 26, 2020

DAVID R. EBERHART
JAMES K. ROTHSTEIN
DANIEL H. LEIGH
O'MELVENY & MYERS LLP

By: /s/ David R. Eberhart
David R. Eberhart

Attorneys for Plaintiffs
ELASTICSEARCH, INC. and
ELASTICSEARCH B.V.